A CRM Application to Handle the Clients and their property Related Requirements

- A Surya Daniel

Project Overview

Dreams World Properties integrates Salesforce to streamline customer interactions. Website engagement triggers automated record creation in Salesforce, capturing customer details and preferences. Salesforce categorizes users as approved or non-approved, offering tailored property selections to approved users. This enhances user experience and efficiency, providing personalized recommendations and broader listings. Seamless integration optimizes operations, improving customer engagement and facilitating growth in the real estate market.

Objectives

Business Goals:

- **Improve Client Management**: Maintain a comprehensive record of client information, preferences, and interactions.
- **Enhance Customer Satisfaction**: Provide personalized service by quickly addressing client property requirements.
- **Streamline Property Management**: Efficiently handle property listings, viewings, and transactions.
- **Boost Sales Efficiency**: Enable sales teams to track and manage leads and opportunities more effectively.
- **Data-Driven Insights**: Generate reports and insights to understand market trends and client needs.

Specific Outcomes:

- **Faster Client Response Times**: Reduced response times to client inquiries.
- **Increased Conversion Rates**: More effective lead tracking and follow-ups.
- **Improved Customer Retention**: Enhanced relationships with personalized interactions.
- **Increased Productivity**: Sales and service teams spend less time on manual data entry and more on client engagement.
- **Better Market Understanding**: Enhanced analytics for market trends and demand prediction.

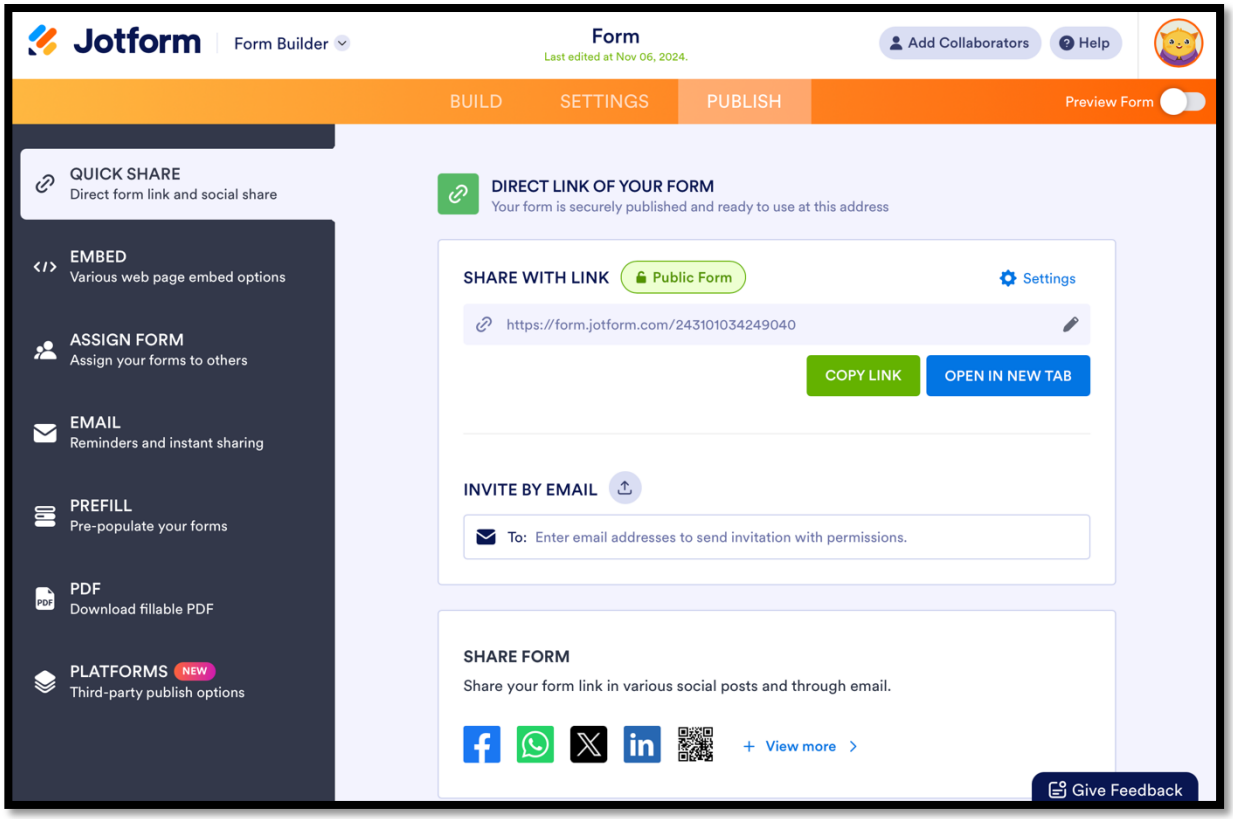
Salesforce Key Features and Concepts Utilized:

- **Sales Cloud**: Manage leads, opportunities, and accounts.
- **Service Cloud**: Track customer inquiries and provide support.
- **Property Management Objects**: Custom objects for properties, listings, and client requirements.
- **Reports and Dashboards**: Visualize key metrics for property and client performance.
- **Mobile Accessibility**: Manage CRM from any device for on-the-go updates.
- **Automation and Workflow Rules**: Automate notifications and follow-up reminders.

Detailed Steps to Solution Design

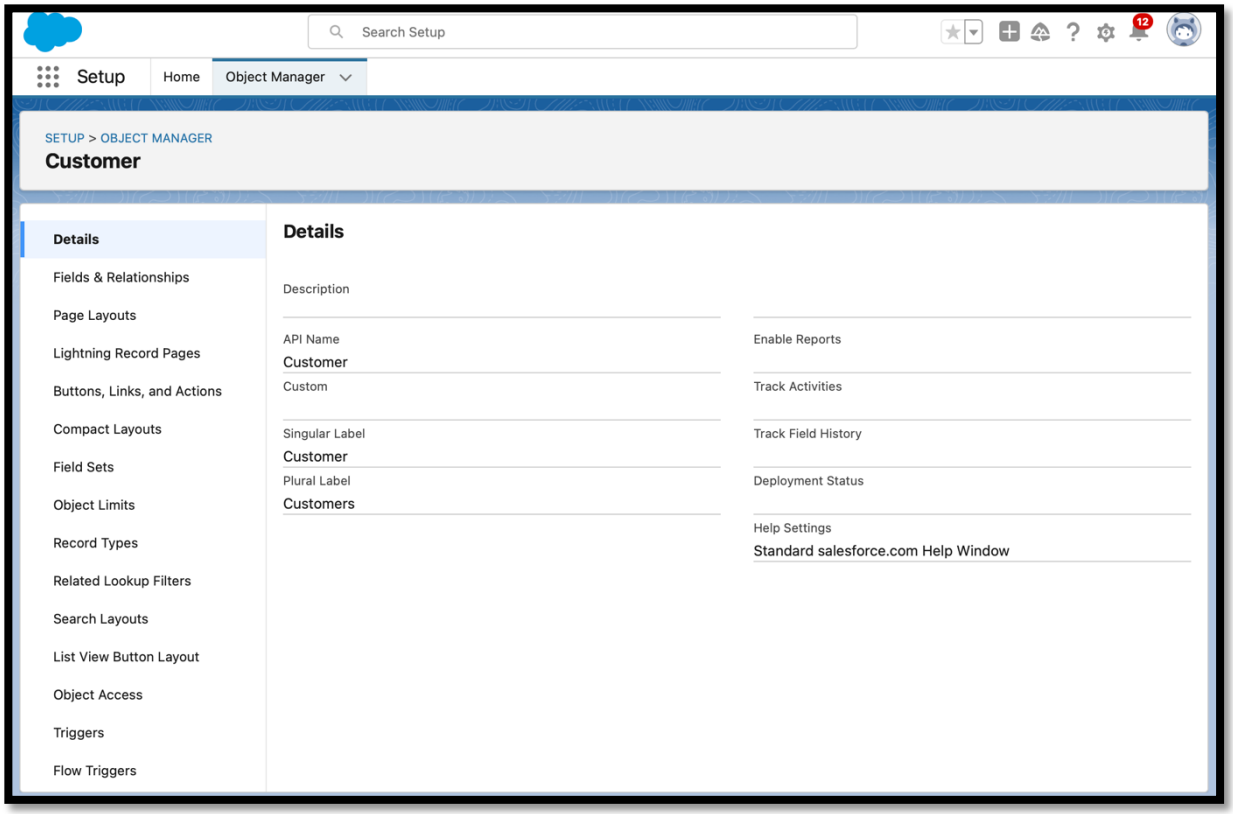
Create a Jotform and integrate it with the org to create a record of customers automatically.

- Open your browser and search for jotform and log in.
- After login click on create form and click on start from scratch
- Now create a form to get the customer details like Name, Phone,Email, Address and type of property the customer is interested in.
- Once the form is created, publish it by clicking on publish.



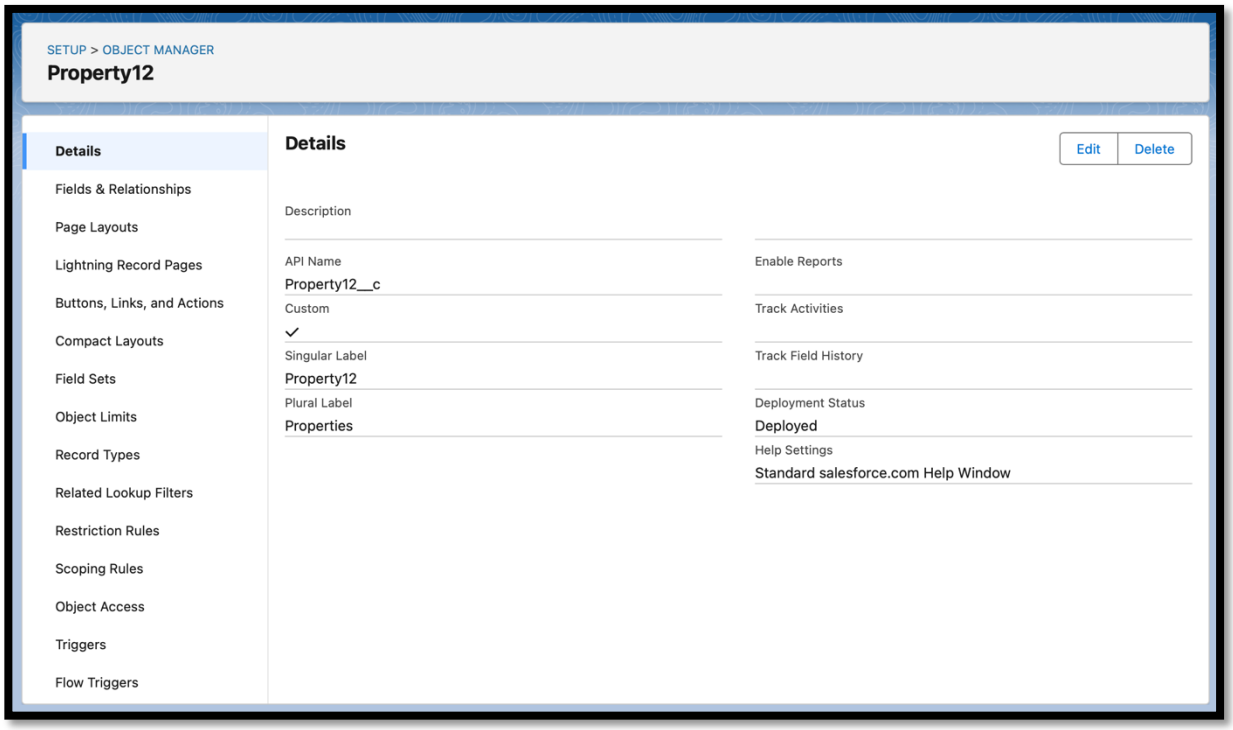
Create Objects from Spreadsheet.
Create Customer Object

- Go to your object manager and and click on create object from spreadsheet.
- Click on the link to get the spreadsheet
- customer
- After downloading, upload the file, map the fields and upload to create an object.



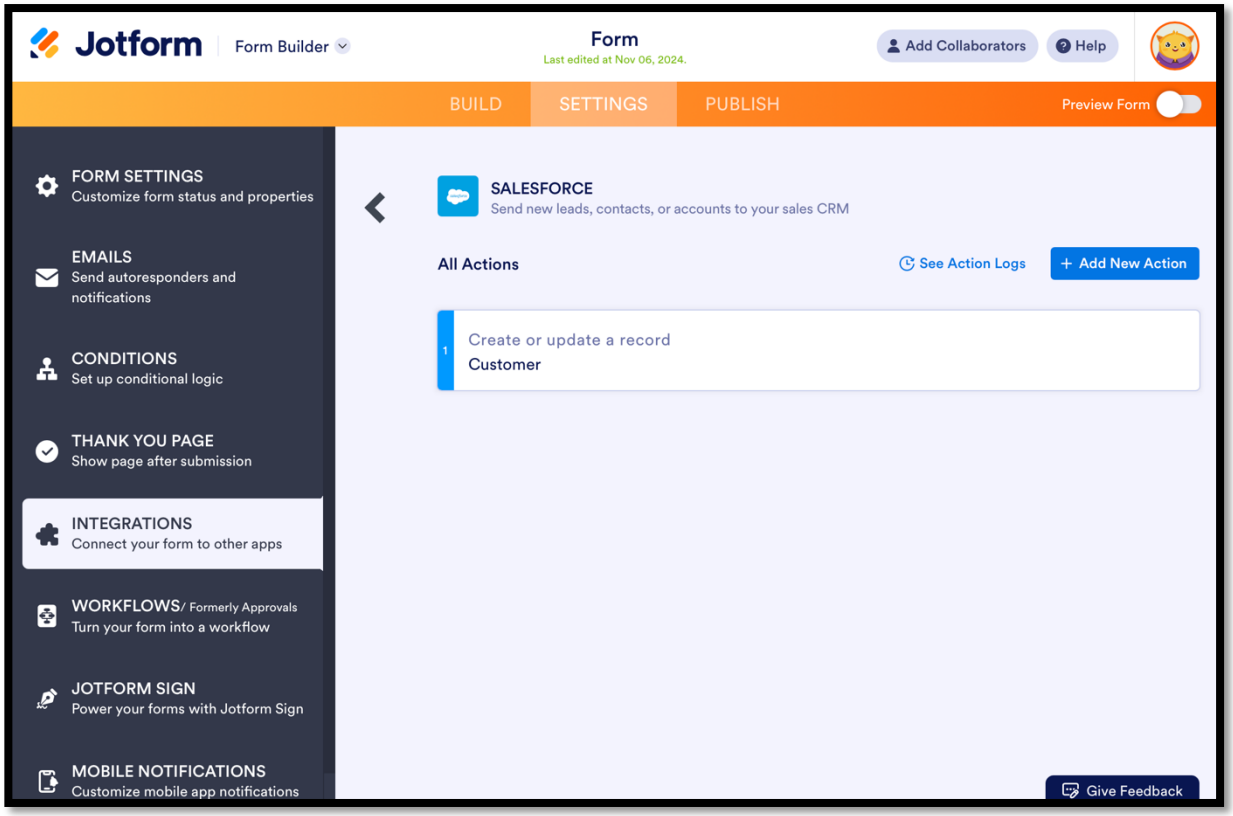
Create Property Object

- Follow the same from the customer object to create the Property Object
- Property



Integrate Jotform With Salesforce Platform

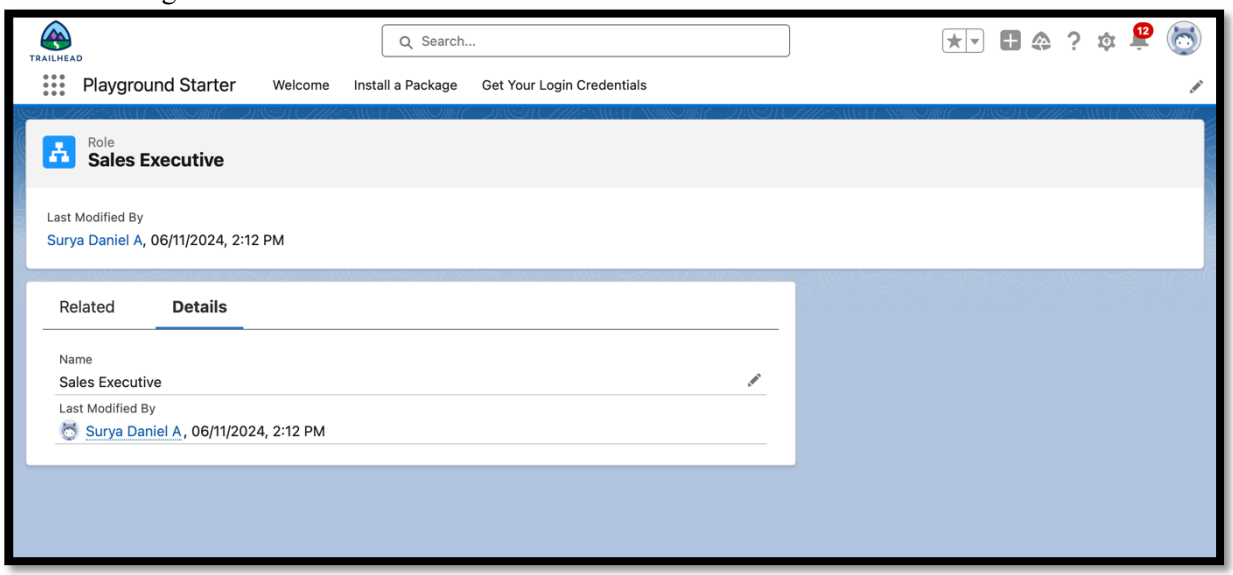
- On the Jotform Platform, Click on Integration and choose Salesforce.
- Click on User Integration and choose “Add to From”.
- Select the Org with which you want to Integrate your jotform with.
- Select an Action - Create a record.
- Select a Salesforce Object : - Customer
- Map Each and every field on the Object with the fields on the form and “Save Action”.
- Then “Save the Integration” and “Finish”.



Create Roles

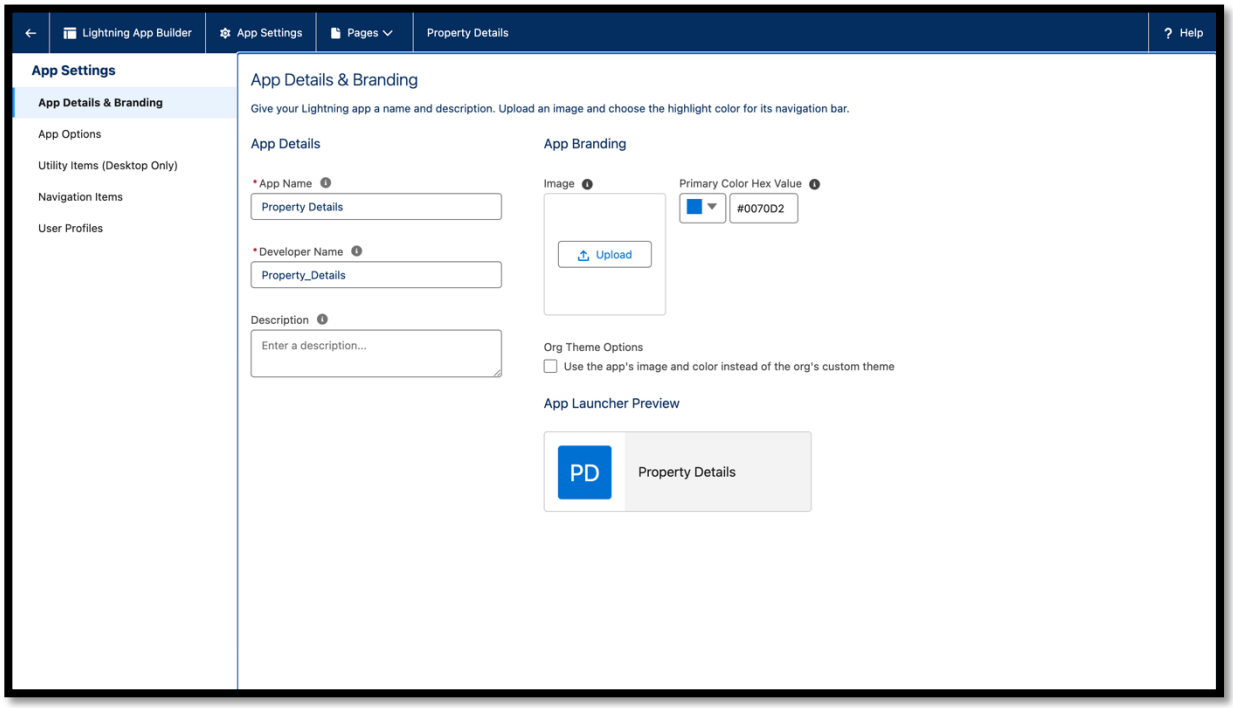
Sales Executive Role

- Go to Setup and Click on Roles, then click on Expand all and Add a Role just below the Sales Representative
- Label - Sales Executive
- Reports to - Sales Representative
- Similarly Create a Role Name “Sales Manager” below Sales Executive which reports to Sales Executive, Also Add a Role below Sales Manager labeled as “Customer” which reports to Sales Manager.



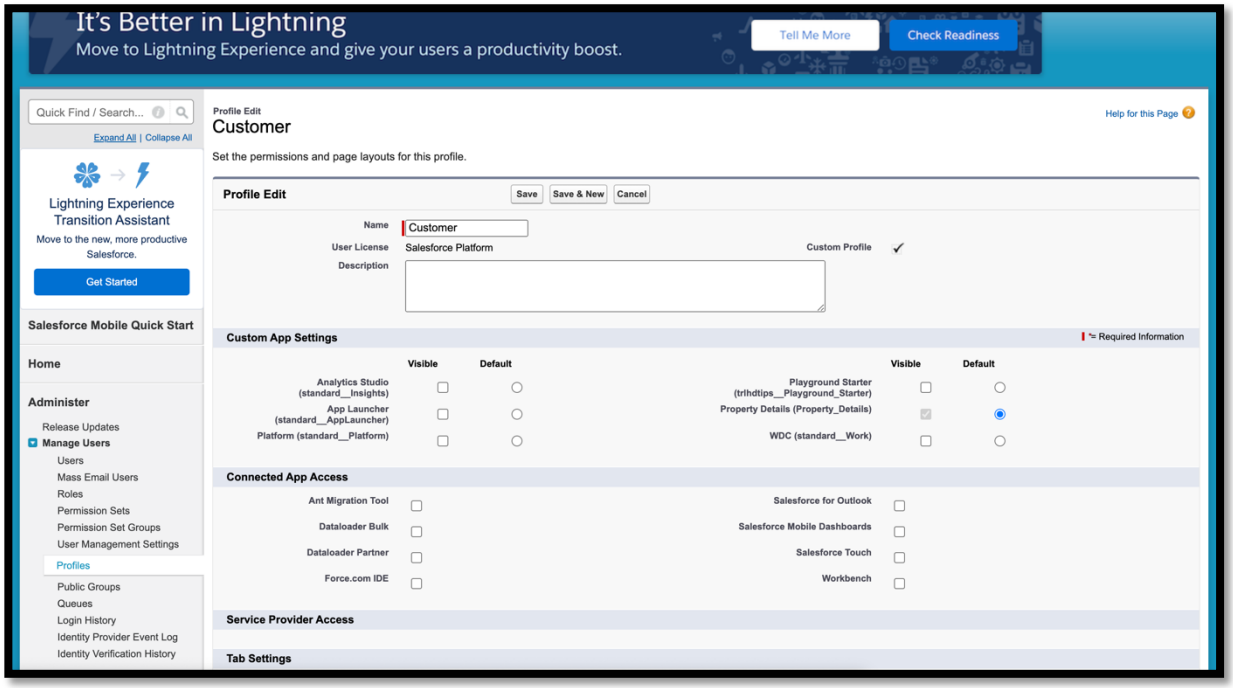
Create A Property Details App

- From Setup>> Go to App Manager and click on New Lightning App and Name it as “Property Details” and add “Customer” and “Property” Object.
- Click Next >> Next >> Save and Add “System Admin ”Profile.



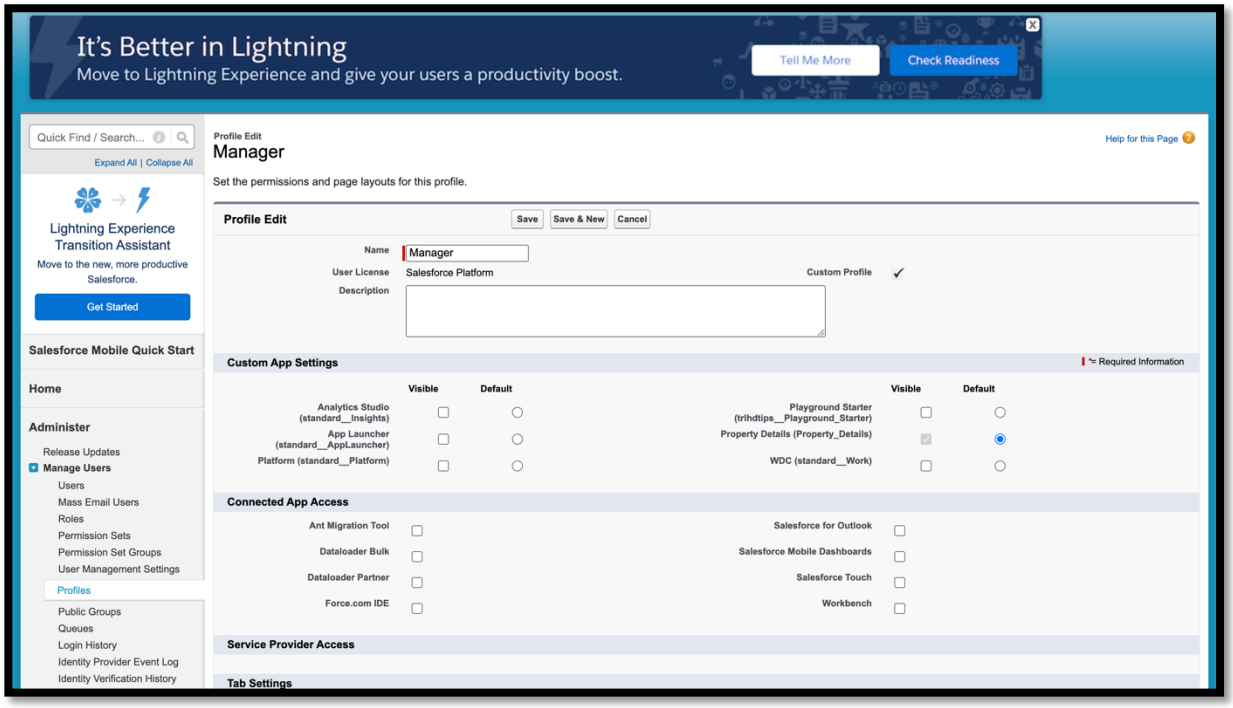
Create Profiles

- From Setup >> Go to Profiles and Clone Salesforce Platform User and Name it “Customer”..
- Uncheck all the Custom Objects and Check only Property Details From Custom App Settings.
- Also Remove all the Standard Object Permissions.
- Uncheck all the Custom Object Permissions and check read and view all in “Property”



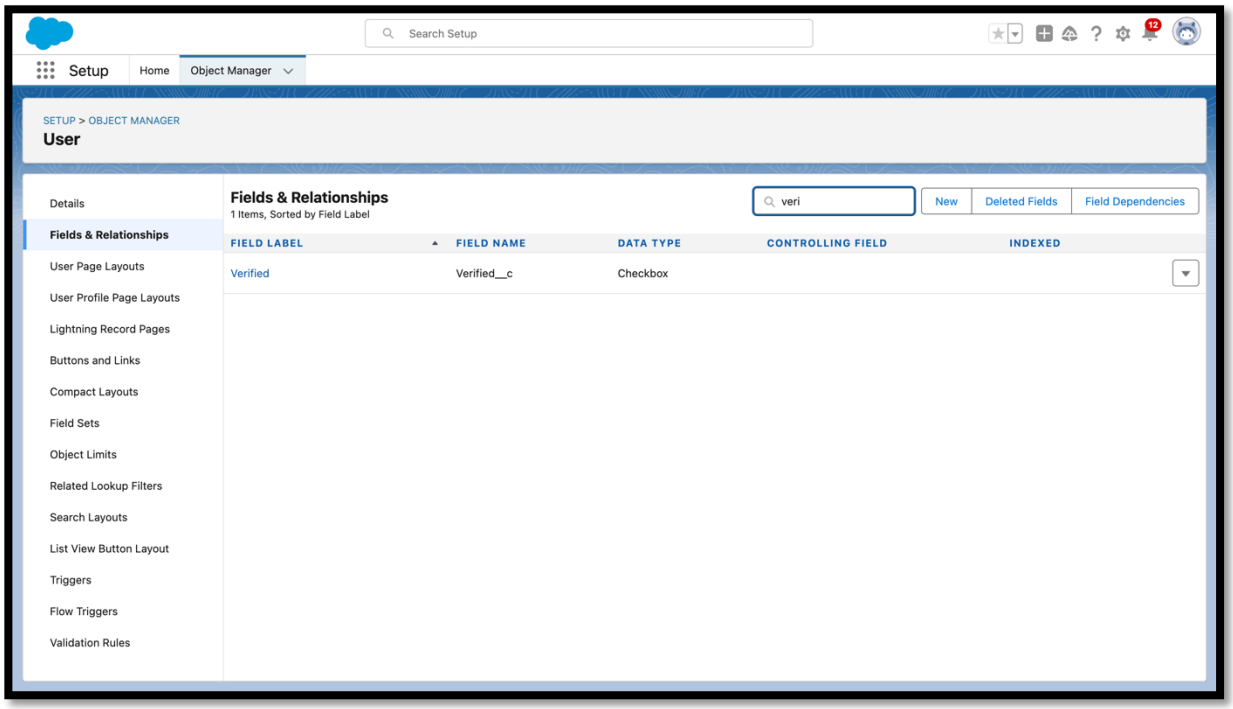
Manager

- From Setup >> Go to Profiles and Clone Salesforce Platform User and Name it “Manager”..
- Uncheck all the Custom Objects and Check only Property Details From Custom App Settings.
- Also Remove all the Standard Object Permissions.
- Uncheck all the Custom Object Permissions and check only “modify all” from “Property” and “Customer”



Create A Check Box Field On User

- Setup >> Object Manager >> Search for User >> Fields and Relationships
- Create new Field Named as “Verified” as Data type “Check Box”



Create Users

User 1

- Go to Setup --> Administration --> Users --> New User
- Last Name - Executive
- Role - Sales Executive
- License - Salesforce
- Profile - System Administrator
- Save

User 2

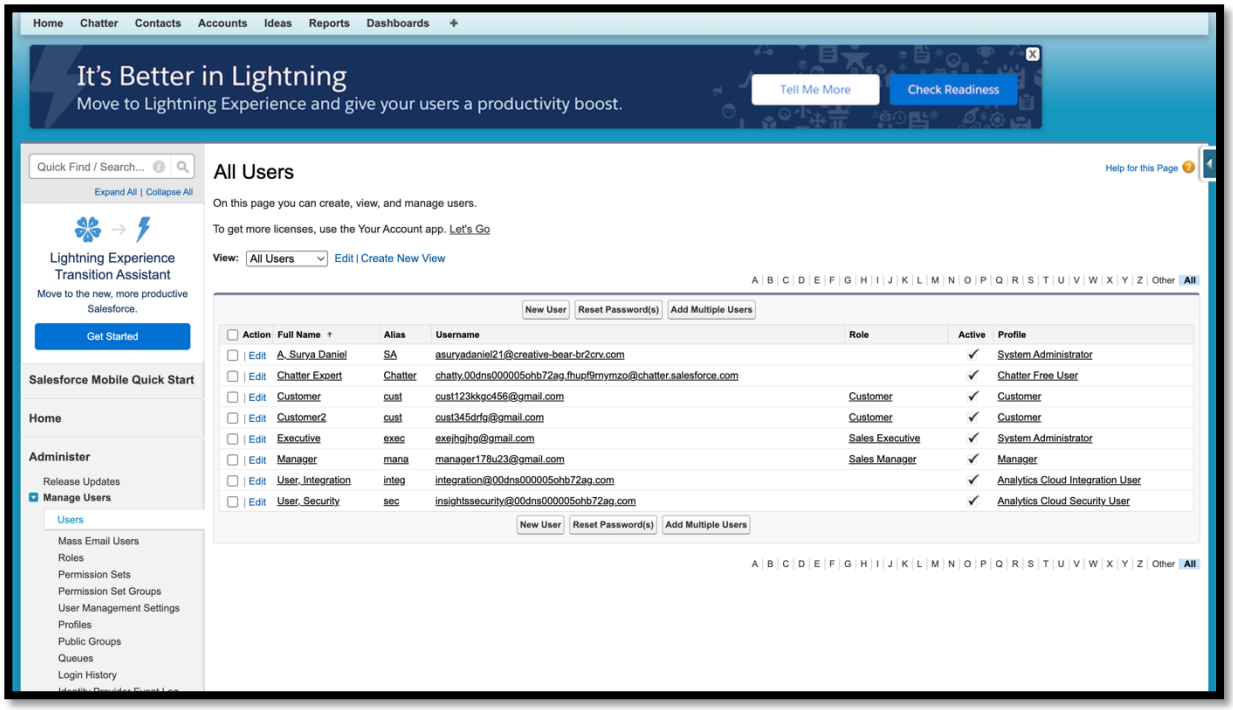
- Go to Setup >> Administration >> Users >> New User
- Last Name >> Manager
- Role >> Sales Manager
- License >> Salesforce Platform
- Profile >> Manager
- Save

User 3

- Go to Setup >> Administration >> Users >> New User
- Last Name >> Customer
- Role >> Customer
- License >> Salesforce Platform
- Profile >> Customer
- Make Sure the verified check box is “Unchecked”
- Save

User 4

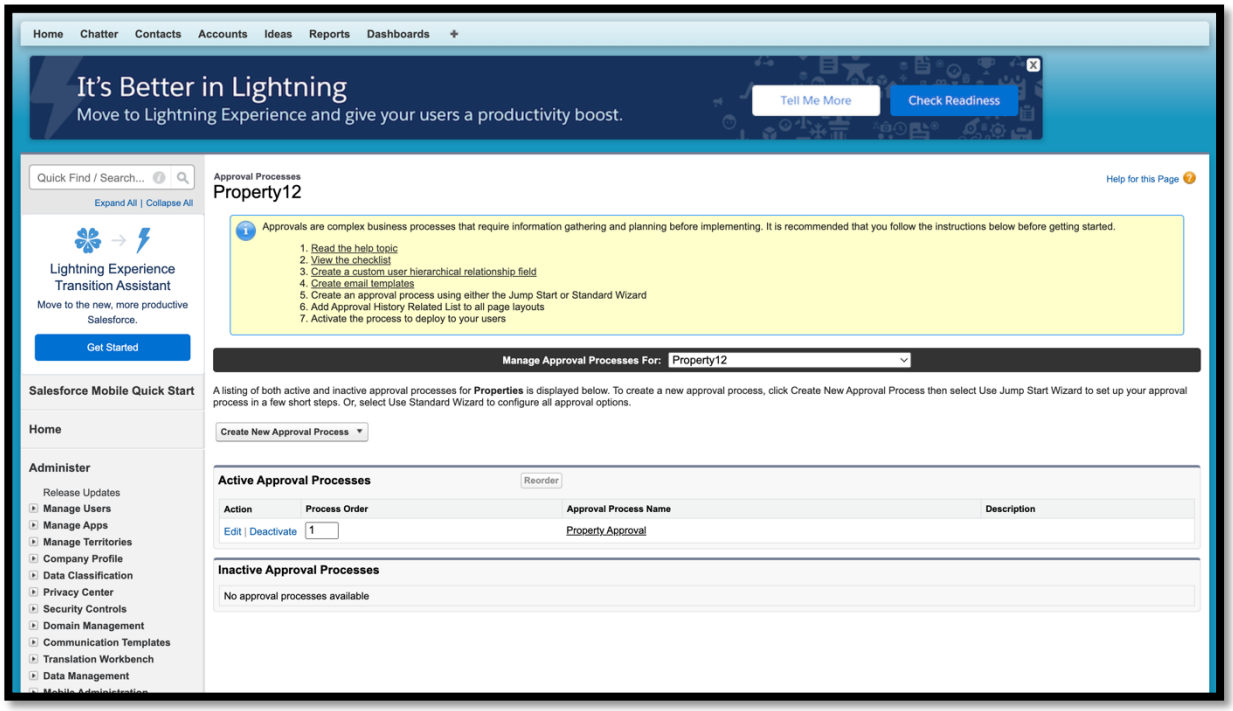
- Go to Setup >> Administration >> Users >> New User
- Last Name >> Customer2
- Role >> Customer
- License >> Salesforce Platform
- Profile >> Customer
- Make Sure the verified check box is “checked”
- Save



Create An Approval Process For Property Object

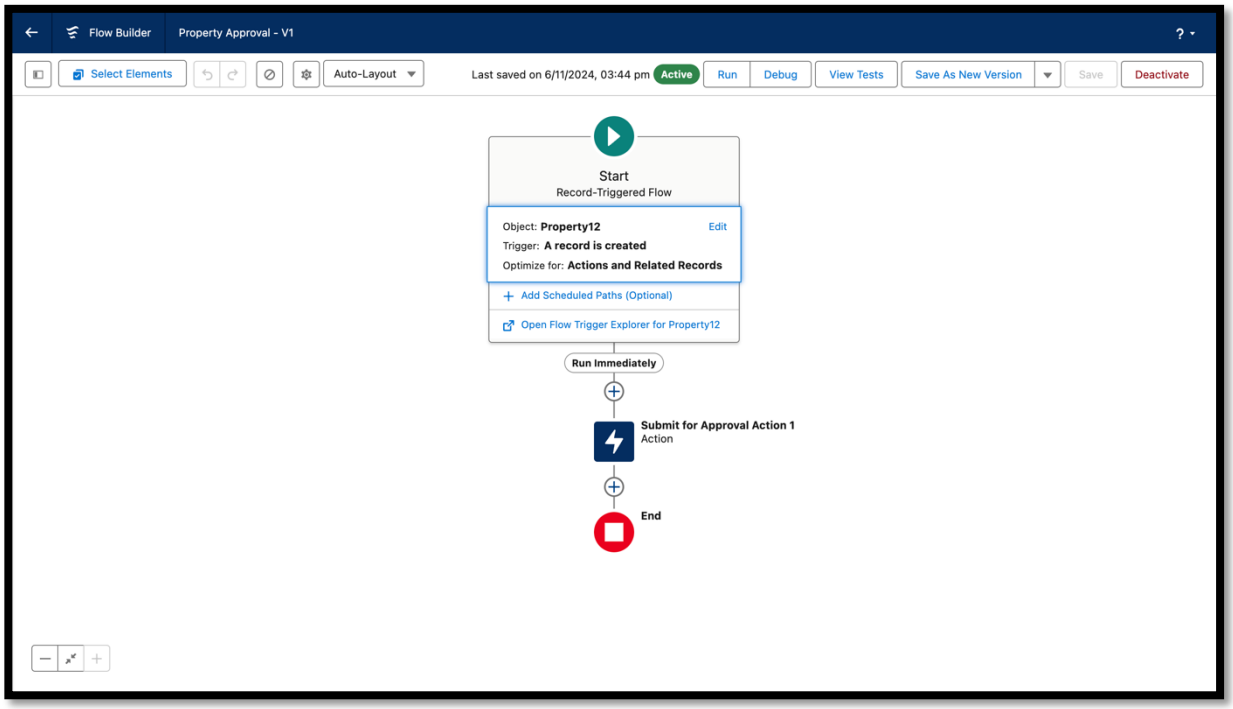
- From Setup >> Process Automation >> Approval Process
- Process Name - Property Approval
- Give 2 criteria –
- Location is not equal to blank,
- Verified Equals false.
- Click next and "Next Automated Approver Determined By" Select Manager
- From Record Editability Properties >> Click on Administrators OR the currently assigned approver can edit records during the approval process.
- From Step 5. Select Fields to Display on Approval Page Layout select Property, Owner, Location, Type.
- Click Next and Select the initial Submiters >>
- Owner >> Property Owner
- Roles >> Sales Manager
- Save.
- Add an approval step name "Executive Approval "
- specify the Criteria >> All record should enter
- click next and select the Approver as " Sales Executive " and "Save"
- Add One field Update as "Verified Property"

- Select Object >> Property
- Field to Update >> Verified
- Field Data Type >> CheckBox
- Select CheckBox Option as “True”
- Save
- Add One field Update as “UnVerified Property”
- Select Object >> Property
- Field to Update >> Verified
- Field Data Type >> CheckBox
- Select CheckBox Option as “False”
- Save.
- Activate the Approval Process.



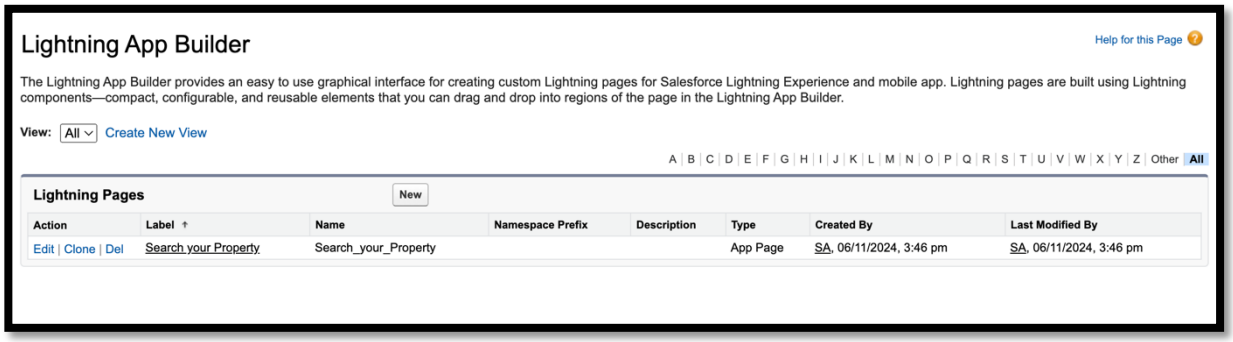
Create a Record Trigger Flow To Submit The Approval process automatically

- From Setup >> Search for Flows >> Click On New and Select “Record Trigger Flow”.
- Select Object >> Property
- Select “Trigger the flow when” >> “A record is created”
- Set Entry Conditions >> “None”
- Add a “Action” >> “Submit for Approval”
- Give Label >> Approval for property
- Record Id >> {!$Record.Id}
- Done
- Save the Flow and Give label as “Property Approval” and “Activate”



Create An App Page

- From Setup >> Go to Lightning App Builder >> Click on New >> Select App Page and Click on Next.
- Give Label as “Search your Property” click “Next”.
- Click “header and Left Sidebar” and Click on “Done”
- Click on “Save ” and then click on “Activate”.
- From Page Setting select page activation as “Activate for all Users”.
- From Lightning Experience Click on “Property Details” and click on Add Page“.
- Then Click on “Save”



Create A LWC Component

- **Create an Apex Class and make it aura enabled and name it “PropertHandler_LWC”**

Code: -

```
public class PropertHandler_LWC{
   @AuraEnabled(cacheable=true)
   public static list<Property__c> getProperty(string type , boolean verified){
      return [SELECT Id, Location__c, Property_Name__c, Type__c, Verified__c FROM Property__c
Where Type__c =: type AND Verified__c =: verified];
   }
}
```

- Create a Lightning Web Component in your VsCode, and (ctrl+shift +P) and click on authorize an org.
- Enter your login id and password to authorize your org.
- Now (ctrl+shift +P) and Create a lightning Web Component and Name it Anything you want to. (Example -)
- **In your Html File Write this code : -**

Code :-

```
<template>
  <lightning-card>
    <div class="slds-box">
      <div class="slds-text-align_left">
        <h1 style="font-size: 20px;"><b>Properties</b></h1>
      </div>
      <div>
        <div class="slds-grid slds-gutters">
          <div class="slds-col slds-size_5-of-6">
            <lightning-combobox name="Type" label="Property Type" value={typevar}
placeholder="Select Property type"
              options={propetyoptions} onchange={changehandler}></lightning-combobox>
          </div>
          <div class="slds-col slds-size_1-of-6">
            <br>
```

```
        <lightning-button-icon variant="neutral" icon-name="standard:search" alternative-text="Search"
          label="Search" onclick={handleClick}></lightning-button-icon>
        </div>
       </div>
     </div>
   </div>
  <template if:true={istrue}>
     <div class="slds-box">
      <lightning-datatable key-field="id" data={propertylist} columns={columns}></lightning-datatable>
     </div>
   </template>
   <template if:false={isfalse}>
     <div class="slds-box">
      <div style="font-size: 15px;"><b>No properties Are Found !!</b></div>
     </div>
   </template>
  </lightning-card>
</template>
```

- **In Your Js File Write this code : -**

Code :-

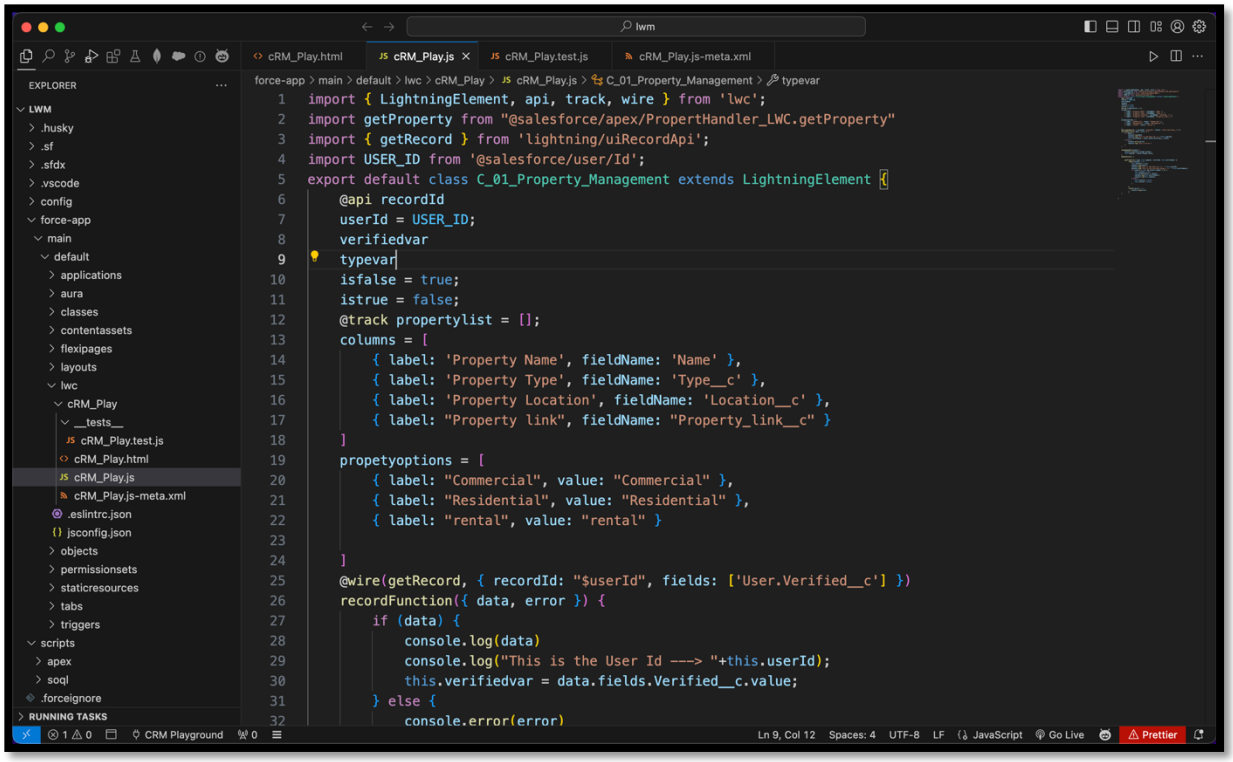
```
import { LightningElement, api, track, wire } from 'lwc';
import getProperty from "@salesforce/apex/PropertHandler_LWC.getProperty"
import { getRecord } from 'lightning/uiRecordApi';
import USER_ID from '@salesforce/user/Id';
export default class C_01_Property_Management extends LightningElement {
   @api recordId
   userId = USER_ID;
   verifiedvar
   typevar
   isfalse = true;
   istrue = false;
   @track propertylist = [];
   columns = [
      { label: 'Property Name', fieldName: 'Property_Name__c' },
      { label: 'Property Type', fieldName: 'Type__c' },
      { label: 'Property Location', fieldName: 'Location__c' },
      { label: "Property link", fieldName: "Property_link__c" }
   ]
   propetyoptions = [
      { label: "Commercial", value: "Commercial" },
      { label: "Residential", value: "Residential" },
      { label: "rental", value: "rental" }

   ]
   @wire(getRecord, { recordId: "$userId", fields: ['User.Verified__c'] })
   recordFunction({ data, error }) {
      if (data) {
         console.log(data)
         console.log("This is the User Id ---> "+this.userId);
         this.verifiedvar = data.fields.Verified__c.value;
      } else {
         console.error(error)
         console.log('this is error')
```

```
    }
  }

  changehandler(event) {
    console.log(event.target.value);
    this.typevar = event.target.value;
  }
  handleClick() {

    getProperty({ type: this.typevar, verified: this.verifiedvar })
      .then((result) => {
        this.isfalse = true;
        console.log(result)
        console.log('This is the User id ---> ' + this.userId);
        console.log('This is the verified values ---> ' + this.verifiedvar);
        if (result != null && result.length != 0) {
          this.istrue = true;
          this.propertylist = result;
          console.log(this.verifiedvar);
          console.log(this.typevar)
        } else {
          this.isfalse = false;
          this.istrue = false;
        }
      })
      .catch((error) => {
        console.log(error)
      })
  }
}
```

- In Your metafile give your targets to deploy the component.
  Code :-

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
  <apiVersion>59.0</apiVersion>
  <isExposed>true</isExposed>
  <targets>
    <target>lightning__RecordPage</target>
    <target>lightning__AppPage</target>
    <target>lightning__HomePage</target>
      </targets>
</LightningComponentBundle>
```

- After Saving all the three Codes , Right Click and deploy this component to the org.



Drag this Component to Your App page

- From Setup >> Go to App Launcher >> Search for Property Details
- On this Page click on gear icon and click on Edit Page
- Drag the Component to your App Page and Save the Page.

Give access on apex classes to profiles

- From Setup >> Search For Apex Classes >> Click on "Security" behind "PropertyHandler__LWC".
- From Profiles Add "Manager" and "Customer" and "Save".